

Supplementary material for the paper *Are Sparse Representations Really Relevant for Image Classification ?* *

Roberto Rigamonti, Matthew A. Brown, Vincent Lepetit
CVLab, EPFL
Lausanne, Switzerland
firstname.lastname@epfl.ch

Abstract

This document presents results we obtained but were unable to put in the paper because of space limitations.

The different combinations we tried are called by concatenating the names of their parts together. The possible part names are given below in capital letters. For example, OLS-SPARSEGD-ABS-MAX-PCA-SVM represents the pipeline where we first extract sparse features computed by Gradient Descent using filters obtained with the Olshausen and Field's algorithm [3], then we use the absolute value function for rectification, use a max-pooling operation, project the result into an eigenspace, and finally use a Support Vector Machine for classification. We sometimes use a star () as the name of one component that is varied for an evaluation.*

1. Filter Bank Learning Setup

The filter learning procedure is based on stochastic gradient descent. The objective function of Eq. (3) is minimized by alternating optimizations over the filters \mathbf{f}^j and the coefficients \mathbf{t}_i^j . Table 1 reports the recognition rates when the filters learned with two extreme values for λ_{learn} , $\lambda_{learn} = 0$ and $\lambda_{learn} = 7$, are used. Throughout the text, the best-scoring configurations will be marked in bold. Also, we will present the ℓ_0 norm of the feature maps both after the feature extraction ($\|\mathbf{t}\|_0$) and after the pooling ($\|\mathbf{v}\|_0$) steps (when relevant).

The corresponding filter banks are depicted in Fig. 1, along with the one obtained by a “proper” λ_{learn} value. Here we report the results for $\lambda_{learn} = 2$ but, as explained in the paper, the same result can be obtained with smaller values, though with a slower convergence rate.

Please note that movies showing the convergence for two different setups are included in the supplementary material.

*This work has been supported in part by the Swiss National Science Foundation.

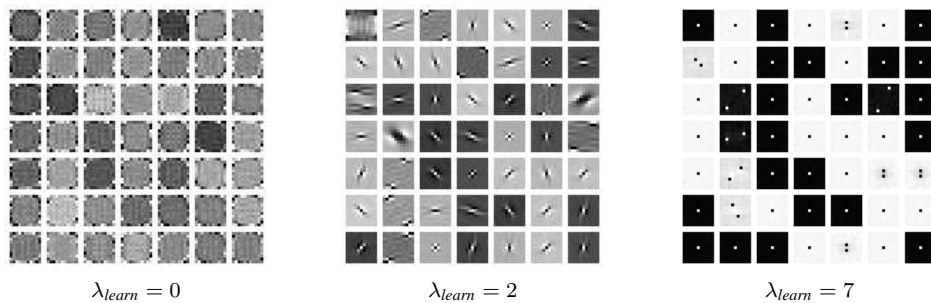


Figure 1: Degenerate filter banks obtained from extreme values of the regularization parameter in the filter bank learning step, along with the one that we actually employed in our tests, corresponding to $\lambda_{learn} = 2$.

Table 1: Comparison of the recognition rates for the filters learned with different λ_{learn} values. These results were obtained on the CIFAR-10 dataset with *OLS- \ast -POSNEG-GAUSS-PCA-SVM*. For a large range of $\lambda_{extract}$ and for a reasonable value for λ_{learn} ($\lambda_{learn} = 2$), the recognition rate is high. For more extreme values for λ_{learn} the performances tend to be worse than the ones obtained with random filters, as shown in Table 4.

Method	$\lambda_{extract}$	$\lambda_{learn} = 0$			$\lambda_{learn} = 2$			$\lambda_{learn} = 7$		
		$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	R. Rate [%]	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	R. Rate [%]	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	R. Rate [%]
<i>CONV</i>		1.000	1.000	55.08	1.000	1.000	67.16	1.000	0.999	54.59
<i>SPARSEGD</i>	1e-4	0.869	1.000	55.91	0.838	1.000	67.28	0.965	0.999	54.11
<i>SPARSEGD</i>	2e-4	0.850	1.000	54.62	0.787	1.000	67.34	0.956	0.999	54.29
<i>SPARSEGD</i>	3e-4	0.829	1.000	56.62	0.741	1.000	67.39	0.945	0.999	54.39
<i>SPARSEGD</i>	4e-4	0.806	1.000	56.55	0.701	1.000	67.30	0.934	0.999	54.28
<i>SPARSEGD</i>	5e-4	0.782	1.000	56.64	0.665	1.000	67.24	0.922	0.999	54.36
<i>SPARSEGD</i>	6e-4	0.757	1.000	56.78	0.633	1.000	67.26	0.911	0.999	54.32
<i>SPARSEGD</i>	7e-4	0.731	1.000	56.83	0.604	1.000	67.26	0.899	0.999	54.20
<i>SPARSEGD</i>	8e-4	0.705	1.000	56.77	0.578	1.000	67.18	0.887	0.999	54.46
<i>SPARSEGD</i>	9e-4	0.679	1.000	56.83	0.555	1.000	67.24	0.876	0.999	54.37
<i>SPARSEGD</i>	1e-3	0.652	1.000	56.82	0.534	1.000	67.11	0.865	0.999	54.36
<i>SPARSEGD</i>	2e-3	0.373	0.996	55.13	0.387	0.998	66.52	0.759	0.998	54.08
<i>SPARSEGD</i>	3e-3	0.182	0.973	54.89	0.287	0.993	65.70	0.642	0.997	53.88
<i>SPARSEGD</i>	4e-3	0.123	0.940	54.48	0.173	0.965	61.69	0.293	0.987	54.11
<i>SPARSEGD</i>	5e-3	0.102	0.912	53.81	0.075	0.883	54.81	0.173	0.978	55.21
<i>SPARSEGD</i>	6e-3	0.088	0.883	53.49	0.060	0.850	53.30	0.150	0.972	55.20
<i>SPARSEGD</i>	7e-3	0.078	0.855	52.38	0.055	0.825	52.55	0.137	0.967	55.00
<i>SPARSEGD</i>	8e-3	0.069	0.827	49.74	0.050	0.802	51.78	0.126	0.963	54.61
<i>SPARSEGD</i>	9e-3	0.062	0.800	51.84	0.046	0.780	51.29	0.117	0.957	54.90
<i>SPARSEGD</i>	1e-2	0.057	0.773	49.12	0.042	0.759	50.28	0.109	0.952	53.55

Table 2: Class abbreviations adopted for the CIFAR-10 dataset.

Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Abbreviation	ai	au	bi	ca	de	do	fr	ho	sh	tr

2. Classification Step Setup

We have identified three configurations for the *SVM* that proved to suit our needs:

- **Fast classification setup.** We used the LIBSVM library, and we have performed a C-Support Vector Classification (C-SVC) by using radial basis functions as kernels, and setting the γ parameter to 10. This is a useful configuration to explore a large parameter space, since classification requires between 1 and 2 hours on a modern laptop.
- **Best performing setup.** We were able to achieve our best results on the CIFAR-10 dataset by solving an expensive multi-class bound-constrained SVC problem with $\gamma = 8$ using the BSVM software [1]. We were unable to systematically adopt this classification scheme since each experiment on our computers required up to a day. The confusion matrices corresponding to the best results are reported in Fig. 3 and Fig. 2 for 16×16 and 32×32 pixels images respectively. The adopted abbreviations are reported in Table 2.
- **Caltech-101 setup.** The tests on the Caltech-101 dataset were performed without any tuning of the SVM. We have just adopted the BSVM implementation bundled with the libHIK library, as reported in the paper, in order to take advantage of the Histogram Intersection Kernel.

ai	0.77	0.02	0.05	0.03	0.03	0.01	0.02	0.01	0.05	0.02
au	0.01	0.84	0.01	0.02	0.00	0.01	0.01	0.00	0.03	0.06
bi	0.05	0.00	0.69	0.06	0.06	0.05	0.03	0.03	0.02	0.01
ca	0.01	0.01	0.06	0.62	0.04	0.14	0.06	0.03	0.01	0.01
de	0.01	0.01	0.08	0.06	0.71	0.04	0.03	0.05	0.00	0.01
do	0.01	0.00	0.06	0.13	0.03	0.68	0.02	0.05	0.01	0.01
fr	0.01	0.01	0.05	0.05	0.04	0.03	0.79	0.01	0.01	0.00
ho	0.01	0.01	0.04	0.03	0.05	0.04	0.01	0.80	0.00	0.00
sh	0.06	0.04	0.01	0.01	0.01	0.00	0.01	0.01	0.81	0.03
tr	0.02	0.06	0.01	0.03	0.01	0.01	0.01	0.02	0.04	0.80
	ai	au	bi	ca	de	do	fr	ho	sh	tr

Figure 2: Confusion matrix for our best scoring configuration, *OLS-CONV-POSNEG-GAUSS-LDE-SVM*, when applied on 32×32 pixels images from the CIFAR-10 dataset. The final recognition rate is 75.28%.

ai	0.79	0.02	0.05	0.02	0.03	0.01	0.01	0.01	0.03	0.02
au	0.02	0.76	0.02	0.03	0.01	0.01	0.02	0.01	0.05	0.08
bi	0.04	0.00	0.67	0.07	0.06	0.04	0.05	0.03	0.02	0.01
ca	0.02	0.02	0.07	0.55	0.06	0.14	0.05	0.04	0.02	0.02
de	0.03	0.01	0.07	0.07	0.63	0.06	0.04	0.07	0.01	0.01
do	0.01	0.01	0.07	0.14	0.06	0.62	0.02	0.05	0.00	0.02
fr	0.01	0.01	0.05	0.07	0.03	0.04	0.74	0.01	0.01	0.02
ho	0.01	0.00	0.03	0.03	0.09	0.05	0.01	0.76	0.01	0.01
sh	0.06	0.05	0.02	0.02	0.01	0.01	0.01	0.00	0.77	0.04
tr	0.04	0.07	0.01	0.04	0.02	0.02	0.01	0.02	0.03	0.73
	ai	au	bi	ca	de	do	fr	ho	sh	tr

Figure 3: Confusion matrix for the *OLS-CONV-POSNEG-GAUSS-LDE-SVM* configuration applied on 16×16 pixels CIFAR-10 images. The results were obtained with the same SVM configuration used for getting the best result on the full resolution dataset, achieving a final recognition rate of 70.49%.

3. Extensive Evaluation of Pipeline Components

We have performed a thorough evaluation aimed at establishing the relative importance of the pipeline’s components, in order to properly tune our classification system.

Since extensive experimentations on the Caltech-101 dataset are prohibitively expensive owing to the resolution of the images, the tests reported below were performed on the CIFAR-10 dataset where the images were resized to 16×16 pixels. Unless otherwise stated, the standard deviation for Gaussian pooling is 2.

Table 3 compares the recognition rates for different non-linearities. The presence of a non-linearity step is important for getting good results [2]. *POSNEG* outperformed *ABS* with both learned and handcrafted filter banks, even though the latter is the one traditionally used in state-of-the-art systems. Despite this result, in the experiments with the Caltech-101 dataset we have adopted the *ABS* nonlinearity, since *POSNEG* has the drawback of doubling the descriptor’s size.

In order to assess the importance of a learned filter bank compared to using a random or an handcrafted filter bank, we performed thorough experimentations adopting different the filter banks in the feature extraction step of our pipeline. Table 4 shows the recognition rate when 49 randomly generated filters are employed, while Table 5 compares in the same way learned

Table 3: Comparison between non-linearities for both learned and handcrafted filter banks. The pipeline configuration is **-CONV-*-GAUSS-PCA-SVM*. *POSNEG* outperformed *ABS* with both learned and handcrafted filter banks.

Method	Rec. Rate [%]	
	<i>OLS</i>	<i>LM</i>
<i>POSNEG</i>	67.16	66.18
<i>ABS</i>	63.17	62.83
<i>NONE</i>	35.61	37.01

Table 4: Comparison of the recognition rates achieved with different $\lambda_{extract}$ values when random filters are used in the extraction stage *RND-*-POSNEG-GAUSS-PCA-SVM*. Random filters work surprisingly well but not as well as learned ones.

Method	$\lambda_{extract}$	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	Rec. Rate [%]
<i>CONV</i>		1.000	0.999	58.13
<i>SPARSEGD</i>	1e-4	0.811	0.999	59.40
<i>SPARSEGD</i>	2e-4	0.802	0.999	59.39
<i>SPARSEGD</i>	3e-4	0.794	0.999	59.39
<i>SPARSEGD</i>	4e-4	0.786	0.999	58.97
<i>SPARSEGD</i>	5e-4	0.778	0.999	59.45
<i>SPARSEGD</i>	6e-4	0.769	0.999	57.28
<i>SPARSEGD</i>	7e-4	0.761	0.999	58.95
<i>SPARSEGD</i>	8e-4	0.752	0.999	58.92
<i>SPARSEGD</i>	9e-4	0.744	0.999	59.55
<i>SPARSEGD</i>	1e-3	0.736	0.999	59.71
<i>SPARSEGD</i>	2e-3	0.654	0.999	58.88
<i>SPARSEGD</i>	3e-3	0.579	0.999	58.26
<i>SPARSEGD</i>	4e-3	0.507	0.999	60.17
<i>SPARSEGD</i>	5e-3	0.434	0.998	57.18
<i>SPARSEGD</i>	6e-3	0.353	0.993	59.65
<i>SPARSEGD</i>	7e-3	0.251	0.970	57.70
<i>SPARSEGD</i>	8e-3	0.130	0.887	52.61
<i>SPARSEGD</i>	9e-3	0.058	0.787	53.02
<i>SPARSEGD</i>	1e-2	0.044	0.747	51.98

and handcrafted filter banks, both using SVM and approximate-NN classification schemes.

Even though the results presented in Table 5 seems to indicate that only little advantage is gained by learning the filter bank compared to using an handcrafted one, the results shown in Table 6 demonstrate that if one takes the “off-the-shelf” Leung-Malik filter bank (that is, it does not alter it by a whitening step), the recognition rate drops below the one achieved by the randomly generated filter bank.

Table 7 lists the results achieved by learned filters on the Caltech-101 dataset when both 15 and 30 training samples are used (as it is usually done in literature).

Matching Pursuit is an algorithm that can be used to obtain feature maps with the desired degree of sparsity. In our experiments, as reported in the paper, it did not succeed in providing an effective representation in terms of recognition rate (see Table 8).

The feature maps obtained by the feature extraction stage can be easily used to provide a reconstruction of the input image. Table 9 shows some of those reconstruction for a given input image and for the different algorithms and filter banks employed, along with the ℓ_2 reconstruction error and the representation’s ℓ_0 norm.

Figure 4 presents an analogous visual inspection for an image taken from the Caltech-101 dataset. Owing to the expensive computations required, we were unable to provide an image for the *SPARSEMP* feature extraction stage.

The introduction of noise in the images enabled us to explore the behavior of the pipeline in a context different from the one usually faced by the systems usually present in literature. We performed our tests on 32×32 pixels images in order to

Table 5: Comparison of the recognition rates between learned (*OLS*) and Leung-Malik (*LM*) filters with whitening, for convolution and different values of $\lambda_{extract}$ and for different classification methods (*SVM* and *NN*) with *-*-*POSNEG-GAUSS-PCA*-. Using sparsity, we can automatically learn filters that perform at least as good as carefully designed ones.

Method	$\lambda_{extract}$	<i>OLS</i>				<i>LM</i>			
				Rec. Rate [%]				Rec. Rate [%]	
		$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	<i>SVM</i>	<i>NN</i>	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	<i>SVM</i>	<i>NN</i>
<i>CONV</i>		1.000	1.000	67.16	44.07	1.000	1.000	66.18	42.69
<i>SPARSEGD</i>	1e-4	0.838	1.000	67.28	45.48	0.569	0.917	66.45	42.99
<i>SPARSEGD</i>	2e-4	0.787	1.000	67.34	45.86	0.505	0.907	66.37	43.71
<i>SPARSEGD</i>	3e-4	0.741	1.000	67.39	45.53	0.458	0.900	66.31	43.46
<i>SPARSEGD</i>	4e-4	0.701	1.000	67.30	45.15	0.418	0.893	66.37	43.65
<i>SPARSEGD</i>	5e-4	0.665	1.000	67.24	45.53	0.382	0.885	65.99	44.21
<i>SPARSEGD</i>	6e-4	0.633	1.000	67.26	45.75	0.349	0.878	66.18	44.13
<i>SPARSEGD</i>	7e-4	0.604	1.000	67.26	45.21	0.320	0.870	65.90	43.66
<i>SPARSEGD</i>	8e-4	0.578	1.000	67.18	45.98	0.294	0.862	65.89	43.82
<i>SPARSEGD</i>	9e-4	0.555	1.000	67.24	45.95	0.275	0.855	66.08	44.58
<i>SPARSEGD</i>	1e-3	0.534	1.000	67.11	45.25	0.259	0.849	65.98	44.68
<i>SPARSEGD</i>	2e-3	0.387	0.998	66.52	45.59	0.169	0.796	64.93	44.66
<i>SPARSEGD</i>	3e-3	0.287	0.993	65.70	45.99	0.124	0.745	63.83	43.89
<i>SPARSEGD</i>	4e-3	0.173	0.965	61.69	44.97	0.097	0.698	62.56	43.65
<i>SPARSEGD</i>	5e-3	0.075	0.883	54.81	44.56	0.078	0.654	61.06	43.99
<i>SPARSEGD</i>	6e-3	0.060	0.850	53.30	44.15	0.064	0.613	59.42	43.97
<i>SPARSEGD</i>	7e-3	0.055	0.825	52.55	44.02	0.054	0.574	58.06	42.73
<i>SPARSEGD</i>	8e-3	0.050	0.802	51.78	43.89	0.046	0.538	56.40	42.35
<i>SPARSEGD</i>	9e-3	0.046	0.780	51.29	43.13	0.040	0.504	54.66	42.87
<i>SPARSEGD</i>	1e-2	0.042	0.759	50.28	43.13	0.034	0.472	53.34	41.55

have a better visual evaluation of the impact of the noise and of the fidelity of reconstruction with the different architectures (see Figure 6). Figure 5 also presents the effect that the different noise types have on the considered test image.

We have systematically analyzed the different components of the pipeline and observed the effects of the different tunings on the final recognition rate. Since the importance of the pooling stage has been discussed in many recent papers, we carefully evaluated the different strategies (see Table 10).

MAX pooling was revealed to be inferior to Gaussian pooling. Table 11 provides a more extensive evaluation for different values of $\lambda_{extract}$. As can be seen, representation’s sparsity is *not* preserved by *MAX* pooling. This result can also be observed in Table 7 for the Caltech-101 case.

To better tune the Gaussian pooling step, we explored different σ values. The results are reported in Table 12.

Finally, we investigated several solutions for the subspace projection step, and we reported the results in Table 13. The number after *RP* represents the cardinality of the final descriptor. Since *RP* achieves results that are not excessively bad compared to *PCA* and *LDE* but with a fraction of the computational cost, we also evaluated the dependence of the recognition rate on the final descriptor’s size (see Table 14).

References

- [1] C. W. Hsu and C. J. Lin. A Comparison of Methods for Multiclass Support Vector Machines. *TNN*, 2002.
- [2] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What Is the Best Multi-Stage Architecture for Object Recognition? In *ICCV*, 2009.
- [3] B. A. Olshausen and D. J. Field. Sparse Coding With an Overcomplete Basis Set: A Strategy Employed by V1? *VISR*, 1997.

Table 6: Comparison of the recognition rates achieved with different $\lambda_{extract}$ values when non-whitened Leung-Malik filters are used (*LM*-*-*POSNEG-GAUSS-PCA*-*). The performances are much worse than the ones obtained with the whitened filters.

Method	$\lambda_{extract}$	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	Rec. Rate [%]	
				SVM	NN
<i>CONV</i>		0.999	0.997	55.52	37.03
<i>SPARSEGD</i>	1e-4	0.610	0.840	50.54	30.85
<i>SPARSEGD</i>	2e-4	0.595	0.838	51.61	30.93
<i>SPARSEGD</i>	3e-4	0.583	0.837	51.33	30.78
<i>SPARSEGD</i>	4e-4	0.572	0.836	51.43	31.01
<i>SPARSEGD</i>	5e-4	0.562	0.834	52.12	30.79
<i>SPARSEGD</i>	6e-4	0.554	0.833	52.17	30.48
<i>SPARSEGD</i>	7e-4	0.545	0.832	51.85	30.87
<i>SPARSEGD</i>	8e-4	0.538	0.831	52.26	30.72
<i>SPARSEGD</i>	9e-4	0.531	0.831	52.16	30.99
<i>SPARSEGD</i>	1e-3	0.524	0.830	52.37	30.81
<i>SPARSEGD</i>	2e-3	0.474	0.823	52.02	30.85
<i>SPARSEGD</i>	3e-3	0.438	0.816	52.92	30.87
<i>SPARSEGD</i>	4e-3	0.410	0.809	52.88	31.36
<i>SPARSEGD</i>	5e-3	0.386	0.801	53.79	31.33
<i>SPARSEGD</i>	6e-3	0.365	0.792	53.73	30.99
<i>SPARSEGD</i>	7e-3	0.347	0.783	53.63	31.12
<i>SPARSEGD</i>	8e-3	0.331	0.774	53.58	30.93
<i>SPARSEGD</i>	9e-3	0.316	0.763	53.75	31.50
<i>SPARSEGD</i>	1e-2	0.302	0.753	53.39	31.92

Table 7: Comparison of the recognition rates achieved with different $\lambda_{extract}$ values when learned filters, initialized with random noise, are used on the Caltech-101 dataset with 30 and 15 training samples (*OLS*-*-*ABS-MAX-SPM-SVM*). Feature maps obtained with plain convolution outperform sparsified ones.

Method	$\lambda_{extract}$	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	Rec. Rate [%]	
				30 samples	15 samples
<i>CONV</i>		1.000	1.000	52.63	43.80
<i>SPARSEGD</i>	1e-4	0.706	0.940	44.36	37.92
<i>SPARSEGD</i>	1e-3	0.488	0.847	46.01	37.70
<i>SPARSEGD</i>	5e-3	0.367	0.649	45.68	37.68
<i>SPARSEGD</i>	1e-2	0.170	0.505	45.05	37.66
<i>SPARSEGD</i>	2.5e-2	0.076	0.285	45.33	38.09
<i>SPARSEGD</i>	5e-2	0.034	0.146	44.13	36.71
<i>SPARSEGD</i>	1e-1	0.014	0.059	41.24	34.76

Table 8: Recognition rate comparison when Matching Pursuit is employed with learned filters on 16×16 images (*OLS-SPARSEMP-POSNEG-GAUSS-PCA-SVM*). Even when the reconstruction error and the sparsity level are similar to those obtained with gradient descent, the recognition rates are significantly lower.

$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	Rec. Rate [%]
0.0010	0.0558	35.6
0.0020	0.0947	35.1
0.0039	0.1548	34.7
0.0077	0.2474	31.2
0.0155	0.3811	32.1
0.0309	0.5553	34.1
0.0618	0.7466	37.2
0.1237	0.9107	41.7
0.2473	0.9947	42.0

Table 9: Images reconstructed from the internal representations obtained by the different filter banks. Sparse representations are very effective in providing a low-error reconstruction with few coefficients.

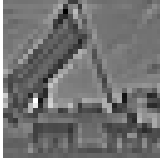
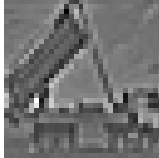
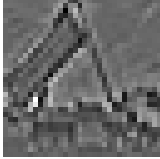
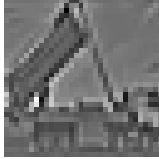
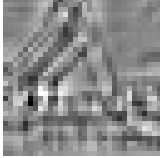
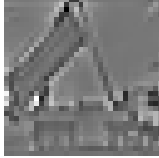
	Original image		<i>OLS-SPARSEMP</i> $\ \mathbf{t}\ _0 = 0.1633$ Reconstruction error=0.00006
	<i>OLS-CONV</i> $\ \mathbf{t}\ _0 = 1.000$ Reconstruction error=0.0021		<i>OLS-SPARSEGD</i> $\ \mathbf{t}\ _0 = 0.195$ Reconstruction error=0.00014
	<i>LM-CONV</i> $\ \mathbf{t}\ _0 = 1.000$ Reconstruction error=0.0464		<i>LM-SPARSEGD</i> $\ \mathbf{t}\ _0 = 0.349$ Reconstruction error=0.0029

Table 10: Comparison between pooling strategies for different subspace projections, (*OLS-CONV-POSNEG-*-SVM*).

Method	Rec. Rate [%]		
	<i>PCA</i>	<i>LDE</i>	<i>RP256</i>
<i>GAUSS</i>	67.16	67.13	66.07
<i>MAX</i>	62.62	61.91	59.92
<i>BOXCAR</i>	63.33	63.33	61.33

Table 11: Recognition rates for *OLS-*-POSNEG-MAX-PCA-SVM* on 16×16 pixel images.

Method	$\lambda_{extract}$	$\ \mathbf{t}\ _0$	$\ \mathbf{v}\ _0$	Rec. Rate [%]
<i>CONV</i>		1.000	0.996	62.64
<i>SPARSEGD</i>	1e-4	0.826	0.991	62.98
<i>SPARSEGD</i>	1e-3	0.510	0.983	62.71
<i>SPARSEGD</i>	1e-2	0.163	0.805	57.93
<i>SPARSEGD</i>	1e-1	0.007	0.070	34.55

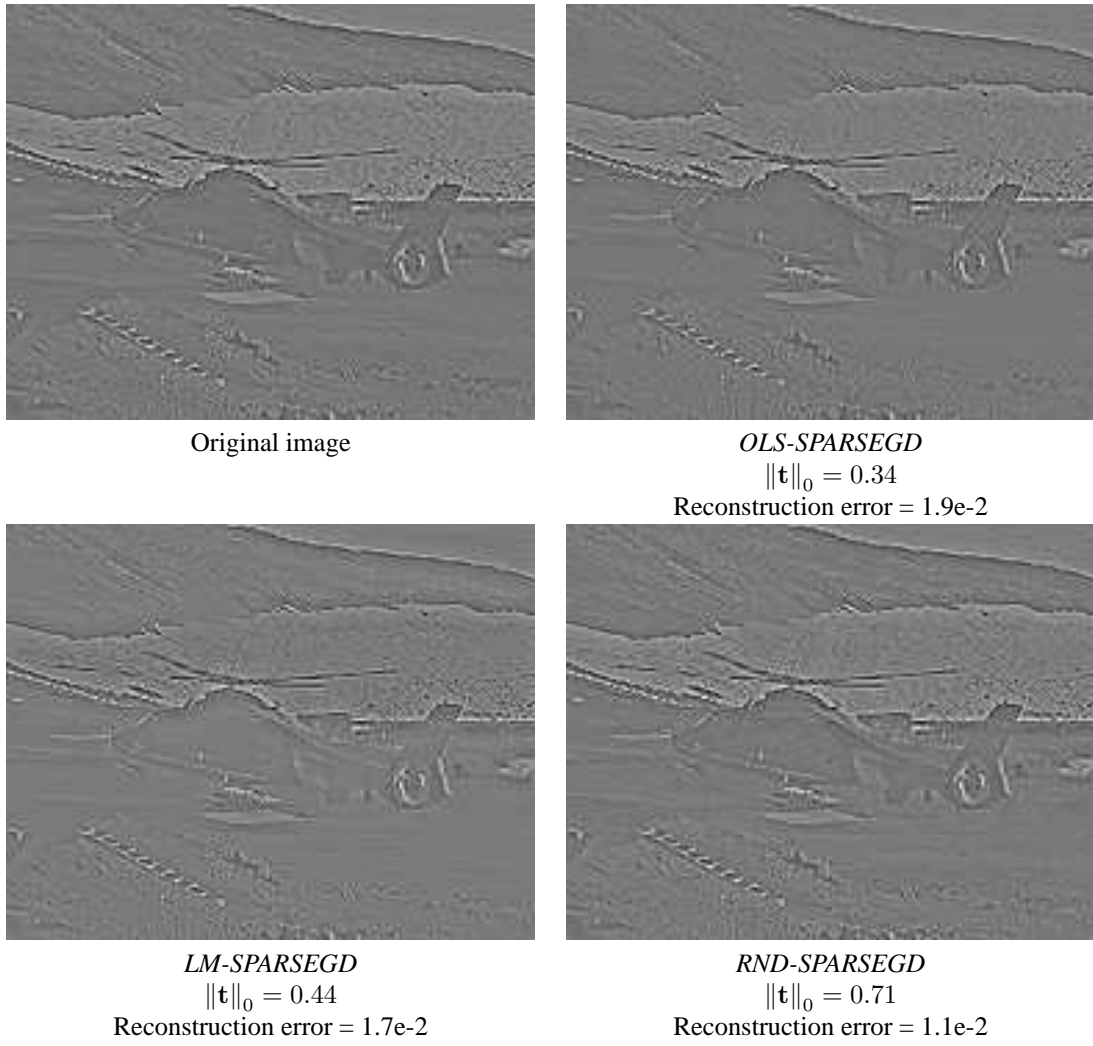


Figure 4: An image taken from the Caltech-101 dataset and whitened, along with the image reconstructed from the \mathbf{t} vectors for different methods.

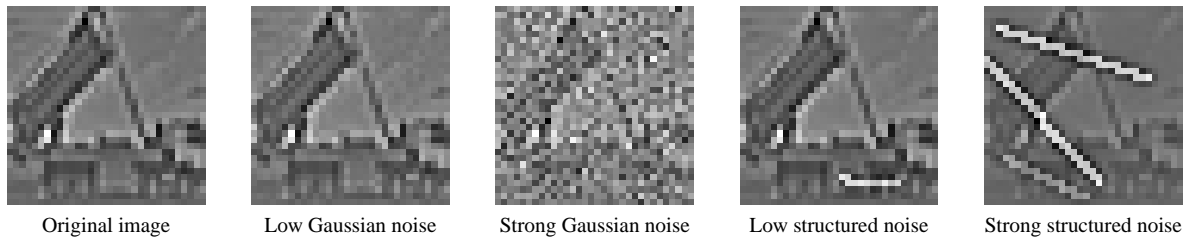


Figure 5: An original image, along with the same image with the different noise types added.

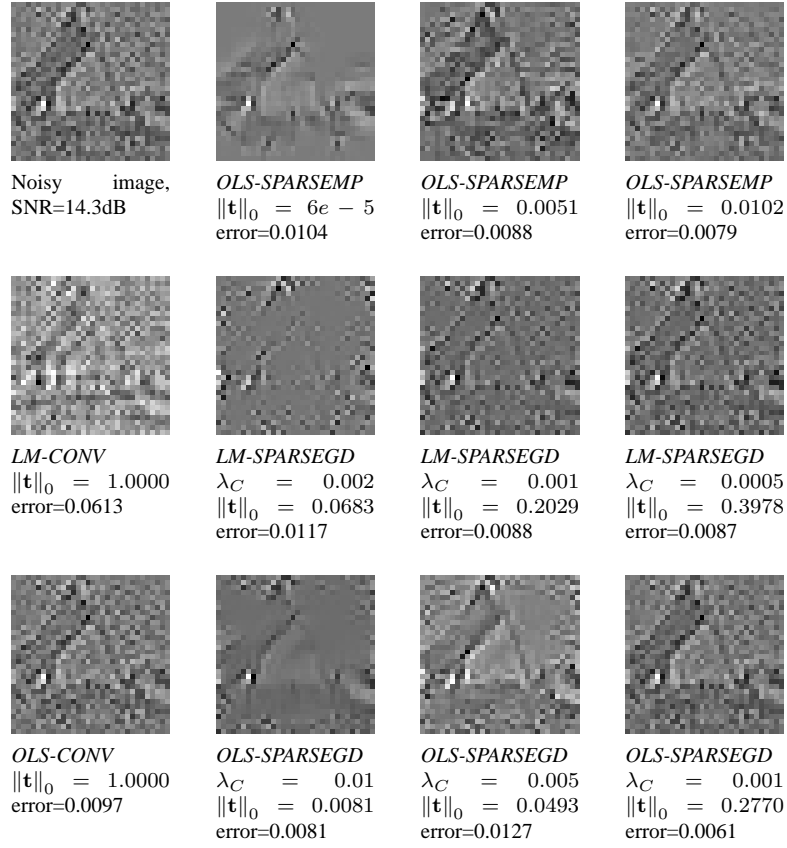


Figure 6: An original image with strong Gaussian noise added, along with the reconstructions obtained by the different architectures for different parametrizations. The error value represents the reconstruction error, the first term of Eq. (4) of the paper.

Table 12: Comparison between different σ values for Gaussian pooling for different subspace projection strategies and nonlinearities (OLS-CONV- σ - σ -SVM). $\sigma = 2.0$ represents a good compromise for POSNEG and ABS.

σ	Subspace Proj.	Rec. Rate [%]	
		POSNEG	ABS
1.0	PCA	66.26	63.93
1.0	LDE	66.26	64.06
2.0	PCA	67.16	63.17
2.0	LDE	67.13	63.62
3.0	PCA	67.20	63.81
3.0	LDE	67.25	63.82
4.0	PCA	67.13	63.83
4.0	LDE	67.14	63.57

Table 13: Comparison between the tested subspace projections, for both learned and handcrafted filter banks and for both *POSNEG* and *ABS* (*-*CONV*-**GAUSS*-**SVM*). *PCA* and *LDE* perform equally well in our experiments.

Method	Rec. Rate [%]			
	<i>OLS</i>		<i>LM</i>	
	<i>POSNEG</i>	<i>ABS</i>	<i>POSNEG</i>	<i>ABS</i>
<i>PCA</i>	67.16	63.17	66.18	62.83
<i>LDE</i>	67.13	63.62	66.34	62.93
<i>RP256</i>	66.07	62.93	64.10	61.26

Table 14: Comparison between the recognition rate when random projections are adopted for the subspace projection step, as a function of the number of retained values (*OLS-CONV-POSNEG-GAUSS*-**SVM*). Keeping 256 dimensions is almost as good as keeping 4 times more.

Method	Rec. Rate [%]
<i>RP64</i>	59.45
<i>RP128</i>	63.08
<i>RP256</i>	66.07
<i>RP512</i>	66.26
<i>RP1024</i>	66.54